<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.DD\_OnfCoreIm-DataDictionaryITUT.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreModel.uml' element=’{0}’ importedBundles='gmf;papyrus' searchMetamodels='true'/>

<gendoc><drop/>

Change path substrings above from “{path for output files}\” to your local path for the output files and “{path for CoreModel}\” to your local path for the Core Model. <drop/>

DELETE: Prior to publishing this –gd.docx (including for review), change path substrings above from “C:\Users\ndavis\git\OnfInfoModelOutput\” to “{path for output files}\” and from “C:\Users\ndavis\git\ONFInfoModel\OnfModel\” to “{path for CoreModel}\” <drop/>

[for (p:Package | self.ownedElement->filter(Package)->sortedBy(name))]<drop/>

## [p.name/]

[p.displayInfo(3)/]

[for (p2 :Package| p.ownedElement->filter(Package)->sortedBy(name))]<drop/>

### [p2.name/]

[p2.displayInfo(4)/]

[for (p3 :Package| p2.ownedElement->filter(Package)->sortedBy(name))] <drop/>

#### [p3.name/]

[p3.displayInfo(5)/]

[for (p4 :Package| p3.ownedElement->filter(Package)->sortedBy(name))] <drop/>

##### [p4.name/]

[p4.displayInfo(6)/]

[for (p5 :Package| p4.ownedElement->filter(Package)->sortedBy(name))] <drop/>

###### [p5.name/]

[p5.displayInfo(7)/]

[for (p6 :Package| p5.ownedElement->filter(Package)->sortedBy(name))] <drop/>

[p6.name/]

[p6.displayInfo(8)/]

[for (p7 :Package| p6.ownedElement->filter(Package)->sortedBy(name))] <drop/>

[p7.name/]

[p7.displayInfo(9)/]

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

</gendoc><drop/>

# Display text with headers procedure <drop/>

Display text using the header of the level provided <drop/>

<fragment name=’displayText’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’text1’ type=’String’/><drop/>

<arg name=’text2’ type=’String’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (level=2)]<drop/>

## [text1/] [text2/]

[else] [if (level =3)]<drop/>

### [text1/] [text2/]

[else] [if (level =4)]<drop/>

#### [text1/] [text2/]

[else] [if (level =5)]<drop/>

##### [text1/] [text2/]

[else] [if (level =6)]<drop/>

###### [text1/] [text2/]

[else] [if (level =7)]<drop/>

[text1/] [text2/]

[else] [if (level =8)]<drop/>

[text1/] [text2/]

[else] [if (level =9)]<drop/>

[text1/] [text2/]

[else]<drop/>

**[text1/] [text2/]**

[/if] [/if] [/if] [/if] [/if] [/if] [/if] [/if]<drop/>

</fragment><drop/>

# Display single element attributes procedure <drop/>

Display the attributes of a single element (class, notification, datatype). <drop/>

Element is the element for which attributes will be displayed. <drop/>

Base is the original class. If B inherits from A, to display all attributes of B, <drop/>

the fragment will be called twice: first element=base=B, element=A, base=B <drop/>

<fragment name=’displaySingleElementAttributes’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’base’ type=’uml::Element’/><drop/>

**[for (p:uml::Property|element.eContents()->filter(uml::Property))]<drop/>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [if (not(p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [if(p.lower=p.upper)][p.lower/][else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(p.isReadOnly)]R[else]RW[/if] | [if not(element.getId()=base.getId())]<drop/>  Inherited from [element.getText()/]  [/if]<drop/>  [if (not(p.default.oclIsUndefined()))]<drop/>  Default: [p.default/]  [/if]<drop/>  [for (st:Stereotype | p.getAppliedStereotypes()->sortedBy(name))]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('bitLength')]bitLength: [if p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('unsigned')]unsigned: [if p.getValue(st, oa.name).oclAsType(Boolean) = true][p.getValue(st, oa.name).oclAsType(Boolean)/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('attributeValueChange')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('valueRange')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('partOfObjectKey')][if (not p.getValue(st, oa.name).oclIsUndefined())][if p.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [p.getValue(st, oa.name).oclAsType(Integer)/]   [else][/if][else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains(‘unit’)]units: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('writeAllowed')]writeAllowed: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] | | | |

**[/for]<drop/>**

</fragment><drop/>

# Display single element operation procedure <drop/>

Display the attributes of an operation element (class, notification, datatype). <drop/>

<fragment name=’displaySingleElementOperations’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’base’ type=’uml::Element’/><drop/>

**[for (o:uml::Operation|element.eContents()->filter(uml::Operation))]<drop/>**

|  |  |  |
| --- | --- | --- |
| [o.name/] | [for (st:Stereotype | o.getAppliedStereotypes()->sortedBy(name))]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('bitLength')]bitLength: [if o.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][o.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('unsigned')]unsigned: [if o.getValue(st, oa.name).oclAsType(Boolean) = true][o.getValue(st, oa.name).oclAsType(Boolean)/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('attributeValueChange')]AVC: [o.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('invariant')]isInvariant: [o.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('valueRange')]valueRange: [if (not o.getValue(st, oa.name).oclIsUndefined())][o.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('partOfObjectKey')][if (not o.getValue(st, oa.name).oclIsUndefined())][if o.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [o.getValue(st, oa.name).oclAsType(Integer)/]   [else][/if][else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('support')]support: [o.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('reference')][if (not o.getValue(st, oa.name).oclIsUndefined())][o.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('condition')][if (not o.getValue(st, oa.name).oclIsUndefined())]condition: [o.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains(‘unit’)]units: [if (not o.getValue(st, oa.name).oclIsUndefined())][o.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('writeAllowed')]writeAllowed: [o.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [for (c:Comment | o.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

**[/for]<drop/>**

</fragment><drop/>

# Display all parameters for all operations procedure <drop/>

Display all the operations in an interface<drop/>

<fragment name=’displayAllParametersForAllOperations’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments; displaySingleElementOperations'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if element.eContents()->filter(uml::Operation)->notEmpty()]<drop/>

**[for (o:uml::Operation|element.eContents()->filter(uml::Operation))]<drop/>**

**[if o.ownedParameter->notEmpty()]<drop/>**

Table 1 Parameters for [o.getText()/]

<table> <drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Direction** | **Multiplicity** | **Applied Stereotypes** |

**[for (p:Parameter|o.ownedParameter)]<drop/>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [p.direction/] | [if(p.lower=p.upper)] [p.lower/] [else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (not(p.default.oclIsUndefined()))]<drop/>  Default: [p.default/]  [/if]<drop/>  [for (st:Stereotype | p.getAppliedStereotypes()->sortedBy(name))]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('bitLength')]bitLength: [if p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('unsigned')]unsigned: [if p.getValue(st, oa.name).oclAsType(Boolean) = true][p.getValue(st, oa.name).oclAsType(Boolean)/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('attributeValueChange')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('valueRange')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('partOfObjectKey')][if (not p.getValue(st, oa.name).oclIsUndefined())][if p.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [p.getValue(st, oa.name).oclAsType(Integer)/]   [else][/if][else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains(‘unit’)]units: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('writeAllowed')]writeAllowed: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | |

**[/for]<drop/>**

**</table><drop/>**

**[/if] <drop/>**

**[/for]<drop/>**

[/if]<drop/>

</fragment><drop/>

# Display operations procedure <drop/>

Display all the operations in an interface<drop/>

<fragment name=’ displayAllOperations’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments; displaySingleElementOperations'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if element.eContents()->filter(uml::Operation)->notEmpty()]<drop/>

Table 2 Operations for [element.getText()/]

<table> <drop/>

|  |  |  |
| --- | --- | --- |
| **Name** | **Applied Stereotypes** | **Description** |

[element.displaySingleElementOperations(element) /]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Display interface procedure <drop/>

Display all the Interfaces <drop/>

<fragment name=’displayInterfaces’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllOperations;displayAllParametersForAllOperations;displayText'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>R

[if element.ownedElement->filter(Interface)->notEmpty()]<drop/>

[element.displayText(‘Interfaces’, ‘’,level)/]<drop/>

[for (i:uml::Interface | element.ownedElement->filter(Interface) ->sortedBy(name))]<drop/>

[element.displayText(i.name, ‘interface’, level+1)/]<drop/>

**Qualified Name:** [i.qualifiedName/]

[i.displayComments()/]<drop/>

[i.displayStereotypes()/]<drop/>

[i.displayAllOperations()/]<drop/>

[i.displayAllParametersForAllOperations()/]<drop/>

[/for] <drop/>

[/if]<drop/>

</fragment><drop/>

# Display comments procedure <drop/>

Display all the comments of the element <drop/>

<fragment name=’displayComments’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

**Description:**

[for (co:uml::Comment | element. ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

</fragment><drop/>

# Display stereotypes procedure <drop/>

Display a list of stereotypes <drop/>

Only for simple stereotypes, not for classes and notifications <drop/>

<fragment name=’displayStereotypes’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if (element.getAppliedStereotypes()->notEmpty())]<drop/>

**Applied Stereotypes:**

[for (st:Stereotype | element.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[for(oa:Property|st.ownedAttribute)]<drop/>

* + [if oa.name.contains('bitLength')]bitLength: [if element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('unsigned')]unsigned: [if element.getValue(st, oa.name).oclAsType(Boolean) = true][element.getValue(st, oa.name).oclAsType(Boolean)/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('attributeValueChange')]AVC: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

* + [if oa.name.contains('invariant')]isInvariant: [element.getValue(st, oa.name).oclAsType(Boolean)/]

[else]<drop/>

* + [if oa.name.contains('valueRange')]valueRange: [if (not element.getValue(st, oa.name).oclIsUndefined())][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('partOfObjectKey')][if (not element.getValue(st, oa.name).oclIsUndefined())][if element.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [element.getValue(st, oa.name).oclAsType(Integer)/]

[else][/if][else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('support')]support: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

* + [if oa.name.contains('reference')][if (not element.getValue(st, oa.name).oclIsUndefined())]reference: [element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('condition')][if (not element.getValue(st, oa.name).oclIsUndefined())]condition: [element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('target')]target: [element.getValue(st, oa.name)/]

[else]<drop/>

* + [if oa.name.contains(‘unit’)]units: [if (not element.getValue(st, oa.name).oclIsUndefined())][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains(‘multiplicity’)]multiplicity: [if (not element.getValue(st, oa.name).oclIsUndefined())]][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + [if oa.name.contains('writeAllowed')]writeAllowed: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display Stereotypes-Indented procedure <drop/>

Display a list of stereotypes <drop/>

Only for simple stereotypes, not for classes and notifications <drop/>

<fragment name=’displayStereotypesIndented’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if (element.getAppliedStereotypes()->notEmpty())]<drop/>

* + **Applied Stereotypes:**

[for (st:Stereotype | element.getAppliedStereotypes()->sortedBy(name))]<drop/>

* + - [st.name/]

[for(oa:Property|st.ownedAttribute)]<drop/>

* + - * [if oa.name.contains('bitLength')]bitLength: [if element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('unsigned')]unsigned: [if element.getValue(st, oa.name).oclAsType(Boolean) = true][element.getValue(st, oa.name).oclAsType(Boolean)/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('attributeValueChange')]AVC: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

* + - * [if oa.name.contains('invariant')]isInvariant: [element.getValue(st, oa.name).oclAsType(Boolean)/]

[else]<drop/>

* + - * [if oa.name.contains('valueRange')]valueRange: [if (not element.getValue(st, oa.name).oclIsUndefined())][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('partOfObjectKey')][if (not element.getValue(st, oa.name).oclIsUndefined())][if element.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [element.getValue(st, oa.name).oclAsType(Integer)/]

[else][/if][else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('support')]support: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

* + - * [if oa.name.contains('reference')][if (not element.getValue(st, oa.name).oclIsUndefined())]reference: [element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('condition')][if (not element.getValue(st, oa.name).oclIsUndefined())]condition: [element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains(‘unit’)]units: [if (not element.getValue(st, oa.name).oclIsUndefined())][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains(‘multiplicity’)]multiplicity: [if (not element.getValue(st, oa.name).oclIsUndefined())]][element.getValue(st, oa.name).oclAsType(String).clean()/]

[else][/if]<drop/>

[else]<drop/>

* + - * [if oa.name.contains('writeAllowed')]writeAllowed: [element.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display all attributes procedure <drop/>

Display all the attribute information in a class, a notification or a datatype<drop/>

It includes attributes from parents. Only 2 levels of parent supported <drop/>

<fragment name=’displayAllAttributes’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displaySingleElementAttributes'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if element.eContents()->filter(uml::Property)->notEmpty()]<drop/>

Table 3 Attributes for [element.getText()/]

<table> <drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Multiplicity | Access | Applied Stereotypes |

[element.displaySingleElementAttributes(element) /]<drop/>

[if (element.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[if (parent.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

</table><drop/>

[else] [if (element.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

Table 4 Attributes for [element.getText()/]

<table> <drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Type | Multiplicity | Access | Applied Stereotypes |

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[if (parent.oclAsType(uml::Classifier).general->notEmpty())]

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

[else]<drop/>

[/if]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Display package procedure <drop/>

Display all the information of a package: QName, comments, stereotypes <drop/>

<fragment name=’displayPackage’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayStereotypes,displayInterfaces()'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

**Qualified Name:** [element.oclAsType(Package).qualifiedName/]

[element.displayComments()/]<drop/>

[element.displayStereotypes()/]<drop/>

</fragment><drop/>

# Display constraints procedure <drop/>

Display all the constraints of the element <drop/>

<fragment name=’displayConstraints’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if element.eContents()->filter(uml::Constraint)->notEmpty()]<drop/>

**Applied Constraints:**

[/if]<drop/>

[for (con:uml::Constraint|element.eContents()->filter(uml::Constraint))]<drop/>

* [con.name/]

[for (cone:uml::Element|con.constrainedElement)]<drop/>

[if cone.oclIsTypeOf(Class)]<drop/>

* + ConstrainedClass: <dropEmpty> [cone.oclAsType(uml::Class).name/] </dropEmpty>

[/if]<drop/>

[if cone.oclIsTypeOf(Association)]<drop/>

* + ConstrainedAssociation: <dropEmpty> [cone.oclAsType(uml::Association).name/] </dropEmpty>

[/if]<drop/>

[if cone.oclIsTypeOf(DataType)]<drop/>

* + ConstrainedDataType: <dropEmpty> [cone.oclAsType(uml::DataType).name/] </dropEmpty>

[/if]<drop/>

[if cone.oclIsTypeOf(Enumeration)]<drop/>

* + ConstrainedEnumeration: <dropEmpty> [cone.oclAsType(uml::Enumeration).name/] </dropEmpty>

[/if]<drop/>

[/for]<drop/>

[for (opq:uml::OpaqueExpression|con.eContents()->filter(uml::OpaqueExpression))]<drop/>

[if (opq.name->notEmpty())]<drop/>

* + [opq.name/]
    - Language: <dropEmpty> [opq.\_language/] </dropEmpty>
    - Body: <dropEmpty> [opq.\_body/] </dropEmpty>

[else][/if]<drop/>

[/for]<drop/>

[for (opq:uml::LiteralString|con.eContents()->filter(uml::LiteralString))]<drop/>

[if (opq.name->notEmpty())]<drop/>

* + [opq.name/]
    - Value: <dropEmpty> [opq.value/] </dropEmpty>

[else]<drop/>

* + Value: <dropEmpty> [opq.value/] </dropEmpty>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

</fragment><drop/>

# Display classes procedure <drop/>

Display all the information in a class: comments, stereotypes and attributes <drop/>

<fragment name=’displayClasses’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(Class)->notEmpty()]<drop/>

[element.displayText(‘Classes’, ‘’,level)/]<drop/>

[for (cl:uml::Class | element.ownedElement->filter(Class) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, ‘class’, level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

[if (cl.isAbstract)]<drop/>

**Abstract class**

[/if]<drop/>

[if (cl.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

**Parent class:** [cl.oclAsType(uml::Classifier).general ->asSequence()->first().name/]

[/if]<drop/>

[cl.displayConstraints()/]<drop/>

**Applied Stereotypes:**

[for (st:uml::Stereotype | cl.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[for (oa:uml::Property|st.ownedAttribute->sortedBy(name))]<drop/>

* + [if (not oa.name.contains('base'))] [oa.name.clean()/]: [if (not cl.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition') or oa.name.contains(‘reference’) or oa.name.contains(‘description’) or oa.name.contains(‘name’) or oa.name.contains(‘multiplicity’)] [cl.getValue(st, oa.name).oclAsType(String).clean()/] [else] [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[cl.displayAllAttributes()/]

[/for] <drop/>

[/if]<drop/>

</fragment><drop/>

# Display notifications procedure <drop/>

Display all the information in a notification: comments, stereotypes and attributes <drop/>

<fragment name=’displayNotifications’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(Signal)->notEmpty()]<drop/>

[element.displayText(‘Notifications’, ‘’,level)/]<drop/>

[for (cl:uml::Signal | element.ownedElement->filter(Signal) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, ‘notification’, level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

**Applied Stereotypes:**

[for (st:uml::Stereotype | cl.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[for (oa:uml::Property|st.ownedAttribute)]<drop/>

[if (not oa.name.contains('base'))]<drop/>

* + [oa.name.clean()/]: [if (not cl.getValue(st, oa.name).oclIsUndefined())] [if oa.name.contains('condition') or oa.name.contains(‘reference’)] [cl.getValue(st, oa.name).oclAsType(String)/]

[else] [if oa.name.contains('support’)] [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]

[if oa.name.contains('triggerConditionList')]<drop/>

[for (s : String | Sequence{cl.getValue(st, oa.name)}->flatten().oclAsType(String))]<drop/>

* [s.clean()/]

[/for]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[cl.displayAllAttributes()/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display datatypes procedure <drop/>

Display all the information in datatypes: comments, stereotypes and attributes <drop/>

Valid for primitive, enumeration and standard datatypes <drop/>

<fragment name=’displayDatatypes’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText;displayStereotypes'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(DataType)->notEmpty()]<drop/>

[element.displayText(‘Datatypes’, ‘’,level)/]<drop/>

[for (dt:uml::DataType | element.ownedElement->filter(DataType) ->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(DataType)]<drop/>

[element.displayText(dt.name, ‘datatype’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[dt.displayConstraints()/]<drop/>

[dt.displayAllAttributes()/]

[else][/if] <drop/>

[if dt.oclIsTypeOf(Enumeration)]<drop/>

[element.displayText(dt.name, ‘enumeration’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

**IsLeaf:** [if (dt.isLeaf)] true [else] false [/if]

[dt.displayStereotypes()/]<drop/>

**Contains Enumeration Literals:**

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]

[if not (e.specification.oclIsUndefined())]<drop/>

* + specification: [e.specification.integerValue()/]

[/if]<drop/>

[for (co:Comment | e.ownedComment)]<drop/>

* + <dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

[e.displayStereotypesIndented()/]<drop/>

[/for]

[else][/if] <drop/>

[if dt.oclIsTypeOf(PrimitiveType)]<drop/>

[element.displayText(dt.name, ‘primitive type’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[dt.displayConstraints()/]<drop/>

[else][/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display associations procedure <drop/>

Display all the information in an association: comments, stereotypes and attributes <drop/>

<fragment name=’displayAssociations’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayText;displayStereotypes’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (element.ownedElement->filter(Association)->notEmpty())]<drop/>

[element.displayText(‘Association’, ‘’,level)/]<drop/>

[for (as:uml::Association | element.ownedElement->filter(Association) ->select((not name.oclIsUndefined()))->sortedBy(name))]<drop/>

[element.displayText(as.name, ‘association’, level+1)/]<drop/>

**Qualified Name:** [as.qualifiedName/]

[as.displayComments()/]<drop/>

[as.displayStereotypes()/]<drop/>

[if (as.memberEnd->notEmpty())]<drop/>

Table 5 Member ends for [as.getText()/]

<table> <drop/>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Aggregation | Navigability | Owner | Multiplicity | Type | Applied Stereotypes |

**[for (p:uml::Property|as.memberEnd)]<drop/>**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.aggregation/] | [if (p.isNavigable())]<drop/>  True  [else]<drop/>  False  [/if] | [if (p.ancestors(Association)->first().oclIsInvalid())]<drop/>  Classifier  ([p.ancestors(Class)->first().name/])  [else]<drop/>  Association  ([p.ancestors(Association)->first().name/])  [/if]<drop/> | [if(p.lower=p.upper)][p.lower/][else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (not(p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [for (st:Stereotype | p.getAppliedStereotypes()->sortedBy(name))]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('bitLength')]bitLength: [if p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('unsigned')]unsigned: [if p.getValue(st, oa.name).oclAsType(Boolean) = true][p.getValue(st, oa.name).oclAsType(Boolean)/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('attributeValueChange')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('valueRange')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('partOfObjectKey')][if (not p.getValue(st, oa.name).oclIsUndefined())][if p.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [p.getValue(st, oa.name).oclAsType(Integer)/]   [else][/if][else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains(‘unit’)]units: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('writeAllowed')]writeAllowed: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | | | |

**[/for]<drop/>**

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display abstraction procedure <drop/>

Display all the information in an abstraction: comments, stereotypes and attributes <drop/>

<fragment name=’displayAbstractions’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayText;displayStereotypes’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (element.ownedElement->filter(Abstraction)->notEmpty())]<drop/>

[element.displayText(‘Abstraction’, ‘’,level)/]<drop/>

[for (as:uml::Abstraction | element.ownedElement->filter(Abstraction) ->select((not name.oclIsUndefined()))->sortedBy(name))]<drop/>

[element.displayText(as.name, ‘abstraction’, level+1)/]<drop/>

**Qualified Name:** [as.qualifiedName/]

Client: [as.client.name/]

Supplier: [as.supplier.name/]

[as.displayComments()/]<drop/>

[as.displayStereotypes()/]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display association warnings procedure <drop/>

Display all the information in an association: comments, stereotypes and attributes <drop/>

<fragment name=’displayAssociationWarnings’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayText;displayStereotypes’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (element.ownedElement->filter(Association)->select(name.oclIsUndefined())->notEmpty())]<drop/>

[element.displayText(‘Association Warnings’, ‘’,level)/]<drop/>

[for (as:uml::Association | element.ownedElement->filter(Association) ->select(name.oclIsUndefined()))]<drop/>

[element.displayText(as.memberEnd->first().name +as.memberEnd->last().name, ‘Unnamed association’, level+1)/]<drop/>

**Qualified Name:** [as.qualifiedName/]

[as.displayComments()/]<drop/>

[as.displayStereotypes()/]<drop/>

[if (as.memberEnd->notEmpty())]<drop/>

Table 6 Member ends for [as.getText()/]

<table> <drop/>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Aggregation | Navigability | Owner | Multiplicity | Type | Applied Stereotypes |

**[for (p:uml::Property|as.memberEnd)]<drop/>**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.aggregation/] | [if (p.isNavigable())]<drop/>  True  [else]<drop/>  False  [/if] | [if (p.ancestors(Association)->first().oclIsInvalid())]<drop/>  Classifier  ([p.ancestors(Class)->first().name/])  [else]<drop/>  Association  ([p.ancestors(Association)->first().name/])  [/if]<drop/> | [if(p.lower=p.upper)][p.lower/][else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (not(p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [for (st:Stereotype | p.getAppliedStereotypes()->sortedBy(name))]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('bitLength')]bitLength: [if p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name <> ‘NA’][p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('unsigned')]unsigned: [if p.getValue(st, oa.name).oclAsType(Boolean) = true][p.getValue(st, oa.name).oclAsType(Boolean)/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('attributeValueChange')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('valueRange')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('partOfObjectKey')][if (not p.getValue(st, oa.name).oclIsUndefined())][if p.getValue(st, oa.name).oclAsType(Integer) <> 0]key: [p.getValue(st, oa.name).oclAsType(Integer)/]   [else][/if][else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition: [p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains(‘unit’)]units: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String).clean()/]   [else][/if]<drop/>  [else]<drop/>   * [if oa.name.contains('writeAllowed')]writeAllowed: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | | | |

**[/for]<drop/>**

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display diagrams procedure <drop/>

Display all the diagrams in a package<drop/>

<fragment name=’displayDiagrams’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayText’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.getPapyrusDiagrams()->notEmpty()]<drop/>

[element.displayText(‘Diagrams’, ‘’,level)/]<drop/>

[for (d:Diagram|element.getPapyrusDiagrams())]<drop/>

Figure 1 Diagram [d.name.clean()/]

<drop/>

<image object=’[d.getDiagram()/]’ maxW=’true’><drop/>

</image><drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display info procedure <drop/>

Display all the information in a package: comments, classes, notifications, datatypes and diagrams <drop/>

<fragment name=’displayInfo’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayClasses;displayNotifications;displayDatatypes;displayAssociations;displayDiagrams;displayPackage;displayAbstractions;displayAssociationWarnings'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[element.displayPackage()/]<drop/>

[element.displayDiagrams(level)/]<drop/>

[element.displayAbstractions(level)/]<drop/>

[element.displayAssociations(level)/]<drop/>

[element.displayClasses(level)/]<drop/>

[element.displayDatatypes(level)/]<drop/>

[element.displayInterfaces(level)/]<drop/>

[element.displayNotifications(level)/]<drop/>

[element.displayAssociationWarnings(level)/]<drop/>

</fragment><drop/>